An Architecture for Visualization of Industrial Automation Data

Guillaume Prévost, Jan Olaf Blech, Keith Foster and Heinrich W. Schmidt

RMIT University, Melbourne, Australia

Keywords: Industrial Automation, Cloud-based Services, Data Visualization.

Abstract: We introduce a framework for visualization of data originating from industrial automation devices. Our framework uses cloud-based services to collect data from industrial automation controllers. Clients can subscribe to the data sources and visualize them in accordance with customer needs. Data from industrial automation facilities is associated with formal semantic models, such as a mathematical representation of the material flow in a production plant. The formal models are used to represent interdependencies between entities, their functionality and other descriptive elements. Ultimately this is used in the visualization and for reasoning about systems. In addition to the software framework we describe work on our demonstrator: an example factory with Raspberry Pi-based controllers that are interconnected via standard ethernet technology.

1 INTRODUCTION

Connecting controllers in factories to internet services comes with a variety of benefits with respect to operation and maintenance. Recent trends are frequently summarized under the term Industry 4.0 (Kagermann et al., 2013). Automation controllers communicating with centralized cloud-based services can not only be used for classical supervisory control and data acquisition (SCADA) tasks, but can also be used for services that are orthogonal to SCADA functionality. Such tasks can comprise additional data analytics and visualization functionality (see, e.g., ABB’s service port framework1) as well as remote health monitoring (see, e.g., (Wenger et al., 2016)). In our work, we are particularly interested in remote monitoring, operation and maintenance of production plants. For example supporting mining site operations, e.g., in the Australian outback is a target area.

In this paper, we introduce a cloud-based framework to gather data from controllers and visualize the data using a web-based frontend. While a variety of products already exist for visualizing industrial facilities such as Dassault Systèmes’ Delmia and Enovia (Dassault Systemes Delmia, 2013; Dassault Systemes Enovia, 2013), we focus on the cloud-based architecture and on a more abstract data visualization view. Visualization is based on formal models. In particular, we are interested in models that express spatio-temporal relationships between entities. A variety of formalisms for spatio-temporal models have been developed. More process algebra-like approaches (Caires and Cardelli, 2003; Caires and Cardelli, 2004) can have benefits when investigating concurrency. On the modeling side, our approach is similar to the qualitative predicates of the Region Connection Calculus (RCC) (Bennett et al., 2002) that can express inclusion, neighborhood and similar spatial properties. Furthermore, the cardinal direction calculus (Skiadopoulos and Koubarakis, 2005), the rectangle algebra (Balbiani et al., 1999), and the cross calculus (Van de Weghe et al., 2005) use comparable means of abstracting from concrete geometric objects in models. On the other hand, semantic descriptions of services in the industrial automation area have been discussed (see, e.g., (Loskyll et al. 2011)) as well as ontologies for factory automation (e.g. (Lin and Harding, 2007)).

The main contributions of this paper are the use of semantic data models for industrial automation in combination with our cloud-based visualization platform. Furthermore, a demonstrator combining these technologies is presented. The demonstrator can be used to analyze production plant operations remotely.

Our data modeling language is introduced in Section 2. The cloud-based data visualization platform is presented in Section 3, while the demonstrator is presented in Section 4. A conclusion and future work are presented in Section 5.

---

2 BeSpaceD DATA-MODELS

In our work, semantic models as well as data in production plants is formalized using the BeSpaceD framework (Blech and Schmidt, 2014). We briefly introduce the language then describe the data structures used in the models.

2.1 BeSpaceD

BeSpaceD is a framework for spatio-temporal modeling and reasoning. It comprises:

- A language for modeling spatio-temporal systems and representing data. The language serves as a domain specific language (DSL) and is realized using abstract datatype constructors provided by the Scala programming language. The language comprises logical operators such as conjunctions, disjunctions and implications as well as operators for time and space as basic entities.

- A library-like collection of operations to reason about the BeSpaceD models as well as import and export functionality. Typical operations comprise abstractions and property detection (such as collisions in time and space).

In the past, BeSpaceD was successfully applied to domains such as train systems (Hordvik et al., 2016), industrial automation (Blech et al., 2015) and smart energy systems (Blech et al., 2016).

2.2 Graphs Representing Industrial Plants

Most of our models for production plants are represented as graphs \((L, E)\) comprising a set of locations \(L\) and edges \(E\). Typically \(L\) can refer to machines, sensors and actuators in a plant while the elements of \(E\) represent interdependencies such as connections, material flow, distances, communication channels. Both edges and locations can be annotated. To give a look and feel, we have realized the following constructors for graphs in BeSpaceD/Scala:

```scala
class BeGraphAnnotated[N, A](terms: List[EdgeAnnotated[N, A]]) extends BIGAND[EdgeAnnotated[N, A]](terms)

class EdgeAnnotated[N, A](val source: N, val target: N, val annotation: Option[A]) extends ATOM
```

For example, we have modeled different aspects of our factory demonstrator. In the evaluation of our framework, we are particularly interested in the material flow topology. This represents the expected flow – between sensors and actuators – of material through the factory. We created a specialized subclass of our graph. The following provides a small excerpt of our graph-based formal model. The listing below shows the definition of an edge in a graph and its use in a very small graph definition comprising a set of two edges. The topology does not need to be static, it can change over time. To represent this, we can annotate the graphs with time constraints.

```scala
def edge(s: FestoSensor, t: FestoSensor) =
  EdgeAnnotated(s, t, Some(ProcessSequence))

BeGraphAnnotated[FestoSensor, TemporalFestoConnection] {
  edge(CapDispenser.StackEjectorRetracted, CapDispenser.StackEjectorExtended) ^
  edge(CapDispenser.StackEjectorExtended, CapDispenser.StackEmpty)
}
```

The nodes (e.g. `StackEjectorRetracted`) are objects that uniquely identify a sensor in the demonstrator.

2.3 Sensor Data

In addition to the static nature of the plant models, we use BeSpaceD to treat live sensor data. Sensor data comprises a sensor identifier that should have a corresponding node in the plant model. Furthermore, it is associated with a timestamp and the actual sensor value. For example, we use the following construct is used to specify that the sensor `StackEjectorRetracted` has the value `Obstructed(High)` at a timepoint `1479976418134`.

```scala
INSTATE(StackEjectorRetracted, 1479976418134, Obstructed(High))
```

The long integers for the time point are recording milliseconds since Epoch (12:00am, Jan 1st, 1970).

Sensor data can be sent using the JSON format. The example above is encoded as follows:

```json
{"type": "IMPLIES",
"premise":{"type": "BIGAND", "terms": [
  {"type": "Component",
  "id": "Stack Ejector Retracted"},
  {"type": "TimePoint",
  "timepoint": 1479976418134}]
},
"conclusion": {"type": "Obstructed",
"signal": High}
}
3 CLOUD-BASED REPORTING
AND DATA VISUALIZATION

This section describes the software platform for the visualization of plant data and data-models formalized using the BeSpaceD framework.

3.1 eStoRED

eStoRED is an open source data visualization platform for industrial decision support and risk assessment. Its architecture is shown in Figure 1. It enables the joint-visualization of data from various data sources or workflows. The visualization is realized using a centralized platform, to make sense of the various pieces of data as a whole, and to provide a way of collaboratively telling a meaningful story about the data. The eStoRED reporting tool offers a way to connect to data sources, retrieve data and visualize it along with the possibility to attach metadata. Possible data sources comprise streamed data (such as sensor data delivered over a network connection), web-services, relational databases and file systems. In eStoRED, users can add their own analysis and risk definition assessment, thereby enriching and adding value to the data displayed. This allows building database-backed comprehensive reports. The eStoRED system can handle static data extracted from files or databases as well as live data – such as data coming from sensors – given that there exist a connector to the data source.

3.2 Architecture

In eStoRED, the main entities created by users are called Stories. Stories contain different Elements: Data Elements are the connected elements visualizing data, Input Elements are the analysis parts written by the users. At its core, the eStoRED platform is composed of a web application backed by a relational database, a message broker and a repository of snippets of code for visualizing data, called Vislets. We describe eStoRED’s components and how they interact together:

- On one end, the data sources are the processes, applications and systems that produce the data. They publish data into messages handled by a publish / subscribe system that orchestrates and distributes messages to the processes that have subscribed. The third-party system chosen for this role is RabbitMQ2, an open-source, secure, robust and scalable system for software messaging, using the AMQP protocol3.
- The Java web application is using the Spring MVC framework, Hibernate ORM to map its data model to a MySQL database storing the internal eStoRED data (Stories, Data Elements, Input Elements, etc.). When working on a Story, a user can create Data Elements and define one or more Subscriptions for each of them.
- A Subscription is composed of a subscription expression, the expected format of the data to be received and the snippet of code, called Vislet that will handle and visualize the data once it is received. eStoRED is connected via a REST API to a curated repository of Vislets and can filter them according to some metadata attached to each Vislet. The eStoRED graphical user interface automatically filters the Vislets to only show those that can handle the expected data format.
- The topic subscription mechanism of RabbitMQ is used for subscribing. The mechanism uses routing keys to match publishers and subscribers. The subscription expression defined in eStoRED is used as the RabbitMQ routing key, a sequence of characters up to 255 bytes, defining dot-separated words and allowing the wildcards characters * (star) substituting for exactly one word and # (hash) substituting for zero or more words. This enables a powerful and flexible mechanism to easily create subscription expressions spanning a wide range of data sources. For example: australia.2016.rainfall, australia.2016.*, #.rainfall are valid routing keys.

Data sources can also use this mechanism to subscribe to each other via the messaging system, and this way create data workflows. This is illustrated at the bottom of Figure 1 where Data Source #3 is subscribed to Data Source #2, and Data Source #4 is subscribed to Data Source #3.

Once Data Elements have been defined, whenever a Story is loaded, the following steps happen, as shown in Figure 1: 1. eStoRED retrieves the Story and the Data Elements it contains. 2. It connects to the Vislet repository and retrieves the Vislets defined in the Subscriptions of each Data Element. 3. The web application then generates a web page where the Vislets are included. 4. On the web page, a JavaScript client for RabbitMQ is executed directly into the client’s web browser to subscribe to the expression. 5. When a Data source publishes a message, if a Data Element is subscribed to it, the message broker passes it on, and the Vislet code is called to in-

2https://www.rabbitmq.com

3https://www.amqp.org/
interpret the data contained in the message, and act on it by displaying it or performing specific computation on it.

4 DEMONSTRATOR AND EVALUATION

We have created a factory demonstrator and connected it to our framework. Figure 2 shows an overview of our food-processing factory demonstrator. The conveyor belt circle for pallets in the middle part and the bottling machinery in the lower left of the picture are visible. One of our Raspberry Pi-based controllers is shown in Figure 3. It features a Raspberry Pi including network connectivity as well as IO-boards to communicate with the sensor and actuator world.

The Figure 4 shows how the eStoRED architecture is used in the context of visualizing that demonstrator. The topology of the food processing plant demonstrator is formalized in BeSpaceD as part of the configuration of the program monitoring the plant. It is converted into the JSON format and sent to the message broker.
broker at initialization. Whenever the sensors’ statuses change over time, the sensors send signals to their respective Raspberry Pi-based controller. A program to monitor this is deployed on the Raspberry Pi. After converting these into the BeSpaceD language, the corresponding events are sent to the message broker via a simple AMQP client.

At the other end, a Data Element is created in the eStoRED platform, with two Subscriptions: one for the topology, and one for the sensor events. The specific visualizers are retrieved and loaded into the web browser. Being in the same data element, both visualizers are acting on the same graph visualization. The topology visualizer draws the nodes and edges of the graph representing the process, and the sensors visualizer re-draws the status of the sensors by colouring the nodes whenever they get updated.

Figure 5 shows an eStoRED Data Element which
A timeline control can be observed at the top of the Data Element, which is updated when receiving new sensor data. Since each sensor signal encompasses the exact time when it happened, the visualizer enables scrolling through signals received in the past, using this timeline control. At the bottom of the Data Element are displayed metadata that can optionally be added to AMQP messages as key-value pairs. Here it only shows metadata as an example, but this could be important data such as the factory location or staff responsible for it. To provide a look and feel, Figure 6 shows a larger graph visualized using eStoRED.

An excerpt of the semantic model represented as a topology in the JSON format is shown below:

```json
{ "type": "BIGAND", "terms": [ { "type": "EdgeAnnotated", "source": { "type": "Component", "id": "Stack Ejector Retracted" }, "target": { "type": "Component", "id": "Stack Ejector Extended" }, "annotation": "ProcessSequence" }, { "type": "EdgeAnnotated", "source": { "type": "Component", "id": "Stack Ejector Extended" }, "target": { "type": "Component", "id": "Stack Empty" }, "annotation": "ProcessSequence" }, ... ]
```

Two edges representing material flow are shown with their annotations. Another graphical depiction of a factory element is shown in Figure 7. Only a limited number of edges are shown for readability proposes The dashed lines represent parts of the factory and these correspond to their relative spatial position and size. Squares represent actuators and circles represent sensors.

Some meta data is shown that is used for configuration, debugging and automatic decision support. Of note is the General Purpose Input Output (GPIO) Pin number of the Raspberry Pi-based PLC that is mapped to the actuator that controls and actuator: it extends a stack ejector. The signal mapping defines the binary voltage level (e.g. zero or 24 volts) that the actuators or sensors accept or emit. This relates to sensor states in our model (e.g. [de]activate actuator; [un]obstructed light sensor). Spatial measurements for a tube that holds bottling caps are shown to il-
There are three different qualitative topological aspects that can be distinguished in our factory model. One edge from each aspect is added to the diagram to illustrate them.

- **Material Flow Topology (green)**
  In the example, this edge is asserting that the stack empty sensor becomes obstructed exactly one second before the stack ejector extended sensor becomes unobstructed. In other words, it takes one second to eject the last cap from the stack.

- **Interdependency aspects (blue)**
  This edge is asserting that the stack ejector extended sensor becomes unobstructed between 200 to 300 milliseconds after the stack ejector extend actuator is inactivated (passive). In other words, it takes 200–300 ms for the light sensor to indicate retraction after the actuator starts retracting...
Stack Ejector Retracted
Stack Ejector Extended
Stack Ejector
Tube
Stack Full
Stack Empty
Workpiece Gripped
Loader Picked Up
Loader Pick Up
Loader Drop Off
Loader Dropped Off

Figure 7: Partial model of a factory element.

- Safety aspects (red)
  This edge is asserting a constraint that we want the loader to move to the pick-up position from half a second before to one and a half seconds after the stack ejector starts extending in order to avoid a collision.

5 CONCLUSION

This paper described our cloud-based data collection and visualization framework for industrial automation. We presented the incorporation of spatio-temporal models into the framework and discussed some detailed examples. In addition, we introduced a demonstrator and a visualization application. The cloud-based software framework and the example factory are integrated and serve as a demonstrator platform for our lab. The presented work facilitates monitoring, operation and maintenance of production plants. In particular remote plants such as mining sites in the Australian outback are a targeted application area. Future work will connect additional services to the AMQP server in order to establish a common interchange platform for factory data.

REFERENCES


DS DELMIA V6R2013x – Fact Sheet: 3DEXPERIENCES
of Global Production Systems for all stakeholders in the extended supply chain. Dassault Systèmes 2013.
ENOVIA V6R2013x – Fact Sheet. Dassault Systèmes 2013